What is claimed is:

1. A system for dynamically linking application code created by a programmer into a running operating system kernel, comprising:

an environment library comprising one or more routines for insulating the application code from the operating system environment and for implementing a uniform execution environment;

a build system for constructing a loadable module from the application code and the environment library;

an execution library comprising one or more routines for encapsulating the loadable module within a standard executable program, transparently loading the loadable module into the running operating system kernel, passing arguments to the loadable module, and terminating and unloading the loadable module after receiving a termination signal; and

a build system for constructing the standard executable program from the loadable module and the execution library.

2. The system of claim 1, further comprising an infrastructure library comprising one or more routines executed prior to loading the loadable module into the running operating system kernel and/or after unloading the loadable module from the kernel.

3. The system of claim 1, wherein the execution library includes one or more routines for setting up input/output channels.

4. The system of claim 1, wherein the executable program may be in several files or a single file.

5. A method, comprising:

creating a loadable module;

creating an executable program; and

executing the executable program, wherein the executable program performs a method comprising the steps of:

setting up input/output channels;

inserting the loadable module into an operating system address space, wherein, once the loadable the module is inserted into the operating system address space, the loadable module begins to execute; and

waiting for the loadable module to connect via kernel/user channels and then connecting those kernel/user channels to the input/output channels.

6. The method of claim 5, wherein after the loadable module is inserted into the operating system address space the loadable module performs a method comprising the steps of:

creating kernel/user channels;

creating a thread to execute the application code; and

waiting for the thread to complete.

7. The method of claim 6, wherein the method performed by the loadable module further includes the step of freeing resources after the thread completes.

8. A computer program embodied on a computer readable medium , the computer program comprising:

a computer code segment for insulating the application code from the operating system environment and for implementing a uniform execution environment;

a computer code segment for constructing a loadable module from the application code and the environment library;

a computer code segment for encapsulating the loadable module within a standard executable program, transparently loading the loadable module into the running operating system kernel, passing arguments to the loadable module, and terminating and unloading the loadable module after receiving a termination signal; and

a computer code segment for constructing the standard executable program from the loadable module and the execution library.

9. A for dynamically linking application code created by a programmer into a running operating system kernel, comprising:

means for insulating the application code from the operating system environment and for implementing a uniform execution environment;

means for constructing a loadable module from the application code and the environment library;

means for encapsulating the loadable module within a standard executable program, transparently loading the loadable module into the running operating system kernel, passing arguments to the loadable module, and terminating and unloading the loadable module after receiving a termination signal; and

means for constructing the standard executable program from the loadable module and the execution library.